

# Addresses

**Address**- A location in memory where data is stored.

**Binary**- Base-2: '0' and '1'.  
Each digit is called a "Bit".  
Computers run off of this.

**Hexadecimal**- Base-16:  
Digits: 0 - 9  
Letters: A - F  
(10 - 15)  
Used as addresses for compactness.

## Address

0xFFFFFFFF

0x00000006

0x00000005

0x00000004

0x00000003

0x00000002

0x00000001

0x00000000

## Value

1001 0101

1100 1100

0110 1110

0000 0000

0110 1011

0101 0001

1100 1001

0100 1111

## Pass by Value

0x7ffdfc3b8a90



**Variable1** (Original copy)

0x7ffdfc3b8a91



**Variable1** (First Copy)

## Pass by Reference

0x7ffdfc3b8a90



**Variable2** (Original copy)



**Variable2** (First Copy)

```

1  #include <iostream>
2
3  using namespace std;
4
5
6  void addTen(int); //Specifying passing an integer
7
8
9
10
11 int main() {
12     int variable1 = 0; //Original copy of "variable1" is declared
13
14     cout << "Input a Number: "; cin >> variable1;
15     cout << endl << endl << "variable1: " << variable1 << endl;
16     addTen(variable1);
17     cout << "variable1 Passed: " << variable1 << endl;
18
19
20     return 0;
21 }
22
23
24
25 void addTen(int variable1) { //New copy of "variable1"; Notice its being redeclared in this scope
26     /*
27     This "variable1" is different than "variable1" in the "main()" function.
28     This "variable1" is not in scope of the "main()" function.
29     */
30     variable1 += 10;
31 }

```

```

samevans@DESKTOP-ELS67E4: /mnt/c/projects/CS1-SamEvans02/Extra_Material/Passing_Value_vs_Reference/Pass_by_Value
samevans@DESKTOP-ELS67E4:/mnt/c/projects/CS1-SamEvans02/Extra_Material/Passing_Value_vs_Reference/Pass_by_Value$ make && make run
g++ -c -g -Wall -std=c++17 Pass_by_Value.cpp
g++ -o Pass_by_Value.out *.o
./Pass_by_Value.out
Input a Number: 5

variable1: 5
variable1 Passed: 5
samevans@DESKTOP-ELS67E4:/mnt/c/projects/CS1-SamEvans02/Extra_Material/Passing_Value_vs_Reference/Pass_by_Value$

```

```

1  #include <iostream>
2
3  using namespace std;
4
5
6  void addTen(int&); //Specifying passing an address of an integer
7
8
9
10
11 int main() {
12     int variable1 = 0; //Original copy of "variable1" is declared
13
14     cout << "Input a Number: "; cin >> variable1;
15     cout << endl << endl << "variable1: " << variable1 << endl;
16     addTen(variable1);
17     cout << "variable1 Passed: " << variable1 << endl;
18
19
20     return 0;
21 }
22
23
24
25 void addTen(int& variable1) { //The same copy of "variable1"; Notice the address is what's being passed
26     /*
27     This "variable1" is the same as "variable1" in the "main()" function.
28     This "variable1", while not in scope of the "main()" function, has its address being passed.
29     */
30     variable1 += 10;
31 }

```

```

samevans@DESKTOP-ELS67E4: /mnt/c/projects/CS1-SamEvans02/Extra_Material/Passing_Value_vs_Reference/Pass_by_Reference
samevans@DESKTOP-ELS67E4:/mnt/c/projects/CS1-SamEvans02/Extra_Material/Passing_Value_vs_Reference/Pass_by_Reference$ make && make run
g++ -c -g -Wall -std=c++17 Pass_by_Reference.cpp
g++ -o Pass_by_Reference.out *.o
./Pass_by_Reference.out
Input a Number: 5

variable1: 5
variable1 Passed: 15
samevans@DESKTOP-ELS67E4:/mnt/c/projects/CS1-SamEvans02/Extra_Material/Passing_Value_vs_Reference/Pass_by_Reference$

```